CHAPTER 13

STRUCTURES

INTRODUCTION

In C++ language, custom data types can be created to meet users requirements in 5 ways: *class, structure, union, enumeration and typedef.*

Structures are one of the 2 important building blocks in the understanding of classes and objects. A structure is a collection of data and functions. In other words, we can say, a structure plus related functions make a class.

INTRODUCTION

Technically ,there is no difference between a structure and a class. In fact , a structure is a class declared with keyword **struct** and by default, all members are public in a structure whereas all members are private by default in the class.

STRUCTURES

- Sometimes, some logically related elements need to be treated under one unit. For instance, the elements storing a student's information (e.g., roll no , name, class, marks, grade) need t be processed together under one roof. Similarly, elements keeping a date's information (e.g., day, month, and year) need to be processed together. To handle and serve to such situation, C++ offers structures.
- Thus it can be said that ,
- A C style structures is a collection of variables referenced under one name.

STRUCTURES

- To handle and serve to such situation, C++ offers structures.
- Thus it can be said that ,
- A C style structures is a collection of variables referenced under one name.

STRUCTURES

- The following code fragment shows how to define a structure (say date). The keyword struct tells the compiler that a structure is being defined.

```
Struct date {  short day;
               short month;
               short year;};
```

REFRERENCING STRUTUERE ELEMENTS

Once a structure variable has been defined, its member can be accessed through the use of (.) the dot operator.

Ex. The code assigns 1740 t the year element of birth_date structure variable declared earlier: birth_date.year = 1740;

the structure variable followed by the (.) & name references to that individual structure element.

REFRERENCING STRUTUERE ELEMENTS

The structure members are just treated like other variables. Therefore to print year of birth_date we can write

```
cout<<birth_date.year;
```

Initializing Structure Elements

- ✓ Structure can be initialized separately or jointly . Member of structure senior_student

  can be initialized as separately:

```
senior_student.rolln=01;
senior_student.class=12;
senior_student.marks=50;
senior_student.grade=A
```

Initializing Structure Elements

Or jointly as:

Stutype senior_student = { 01 ,12 ,50 ,A };

Joint structure can not be used befour the variables are defined.

Structure Assignment

- ✓ Objects of the same structure can be assign

  or passed as a function .

Ex. The members of senior_student can be assign to junior_student.

Structure Assignment

Two structure types are different ven they have same members.

```
Struct one {  int a ;
              };
Struct  two{int a;
              };
              one     s1 ;
              two     s2;
              cin >> s1.a;                          //read s1
               s2=s1                                    //error    :type
    mismatch
```

The code will produce an error because both are of different type.

NESTED STRUCTURE

A structure element may be either complex or simple .the simple elements are int, char, float double .

Element of structure may contain structure in itself known as complex structure.

ACCESSING NESTED STRUCTURE MEMBER

The member of structure can be accessed by using dot operator . To access the city member of address which is the element of other structure worker, we shall write:

Worker.address.city

STRUCTURE AND ARRAYS

Structure and arrays are both derived types. Arrays are the collection of analogous elements,

structures assemble dissimilar elements under one roof. Both can be combined to form compels data objects.

ARRAYS OF STRUCTURE

An array can contain similar elements,the combination having structures within an array is an array of structure .To declare an structure tou must define a structure and then declare an array variable of that type.

To declare a 100 element array of structure of type

```
addr  men_addr  [100] ;
```

To access a specific structure ex . to print houseno of structure 8 write

```
cout  <<  mem_add [7].houseno  ;
```

Passing structure to function

Passing a local structure to a function can be done in two ways:

1. By passing individual structure element
2. By passing the entire structure.

Passing structure elements to function

- ✓When an element of structure is passed to a function, actually a value of that element to that function is passing . It is just like passing the simple variable.
- ✓Passing entire structure makes sense when structure is relatively compact.
- ✓The entire function can be passed by value or by reference.

Returning structure from function

- ✓ Just like other types , function can return structure also. Then the type of the function is the same as that of structure returned.

User defined data type

- C++ allows you to define new type of data types name by using keyword typedef
- Typedef does not create new data class rather it define new name for an existing type.
- Ex typedef float amount

#Define Pre-Processor Directive

- ✓ Pre-processor commands are called DIRECTIVES and begins with a pound symbol.(#). Many things that can be done during pre-processor phase include:
- ✓ Inclusion of other files through #include directive.
- ✓ Definition of symbolic constants and macros through #define directive.

#Define Pre-Processor Directive

- The preprocessing phase of a c++ program occurs befour a program is compiled . The c++ peprocessor is a program that is executed befour the source code is compiled.

#Define Pre-Processor Directive

- The preprocessor allows us to define symbolic names and constants. Ex .
- 1. #define I 3.14159
- 2.#define MAX 70

Macros are built on #define preprocessor

A micro define would be

#define SQUARE (x) x*x

#Define Pre-Processor Directive

The difference is of constant and expression.

```
#include <iostream.h>
#define square (x) x*x
Void main()
{int value=3;
        cout << square (value);
}
```

After preprocessing the code would become

```
Void main()
{int value =3;
Cout <<value*value}
```

#Define Pre-Processor Directive

A few things that must be known about macros

- ✓ A macro without argument is treated like symbolic constant.
- ✓ A micro substitutes text only ; it does not check for data type.
- ✓ While defining macros ,make sure you use parenthesis.

```
#define CIRLE_AREA(X) PI*X*X
area =CIRLE_AREA(C+2);
```

IT WOULD BE EXPANDED AS

```
area =3.14159*C+2*C+2;
```